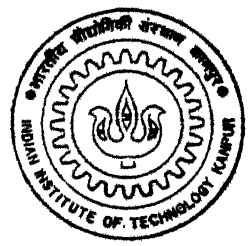
Digital Classroom: A Case Study

by

Lt. Vivek Gupta



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Indian Institute of Technology Kanpur

MARCH, 1998

Digital Classroom: A Case Study

A Thesis Submitted
In Partial Fulfillment of the Requirements
For the Degree of
Master of Technology

By
Lt. Vivek Gupta

to the
Department Of Computer Science & Engineering
Indian Institute Of Technology, Kanpur
March, 1998

CSE-1998-M- GUP - DJG

Entered in the System
N. 3.9
1.6

For my Dad, Mom
And
Monica

Certificate

This is to certify that the work contained in the thesis entitled **Digital Classroom : A Case Study** by **Lt. Vivek Gupta** has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Dr. HC Karnick
Professor
Department of Computer Science & Engineering
Indian Institute of Technology. Kanpur.

Abstract

A tool, **Hyperboard**, a Hypertext based concept, has been designed as a digital aid for classroom instruction by a human Instructor. The tool helps the Instructor in developing and organizing/structuring the information base. A set of Hyperboards result in the generation of a lecture. And in the same manner a set of lectures results in the generation of a course. The Instructor is provided with tools to organize/structure the information in a semantic hypertext network. The students desiring to access the Information space on-line can navigate through the web of information so created.

This thesis is a full case study of building such a course using an authoring system (which is another concurrent Mtech Thesis). This case study deals with a Hyperboard rendition of a Java Programming course.

Acknowledgements

This piece of work would not have found the end of the tunnel had it not been the help of some persons. Let me express my sense of gratitude to my guide /advisor/ mentor Dr HC Karnick, (Words are less to express my gratitude to him). He has shown me the path throughout the journey, which I had to undertake. He lifted me from the nadir point and placed me at the zenith.

I am extremely grateful to Dr TV Prabhakar for material assistance provided throughout the thesis. He was always ready to impart with all available material and supported me in my work.

I am grateful to Ms Ruchi Goel, Mr. Anil Kumar and Mr. Ravi Kiran for extending those helpful hands whenever they were needed most.

I am grateful to my colleague Flt Lt. Davesh Singh for providing me with assistance whenever required. His helpful nature will go a long way with me. Mr. Atul Kumar has played a younger brothers role with his Omni presence whenever required. Thanks to all my colleagues to offer their helping hands at all times.

Special mentions go to my wife and companion " Monica ". Despite of the fact that we were recently married she never complained for our long hours of separation. Her cheerfulness at all times gave me thrust to work with more enthusiasm.

Contents

List of Figures

Chapter 1

Introduction

The power and flexibility that the computers offer make it possible to significantly improve the tools available for traditional classroom teaching. Existing tools include blackboard and chalk, white boards and markers, viewgraph projectors and more recently multimedia presentation hardware and software. There is a very real need to develop conceptual framework software and the requisite software for authoring and presenting classroom material.

1.1 Motivation:

Presentation software currently available has been designed for short one-off presentations. It is not completely suited for traditional classroom teaching.

This thesis work starts with the premise that a human teacher is indispensable in classroom teaching. The focus, therefore, shifts to what kind of digital aids will make classroom teaching more effective.

In order to design the right kind of digital aids we need to understand what exactly goes into the classroom teaching and what is it that a teacher actually does in the process of teaching.

The classroom instruction has three components:

i. Lecture -- emphasis is conceptual.
ii. Assignments and laboratory exercises -- illustration, skill acquisition, manipulative expertise and concepts in practice.
iii. Testing and Examination -- emphasis on differentiation in a population of students based on mastering of material in (i) and (ii) above.

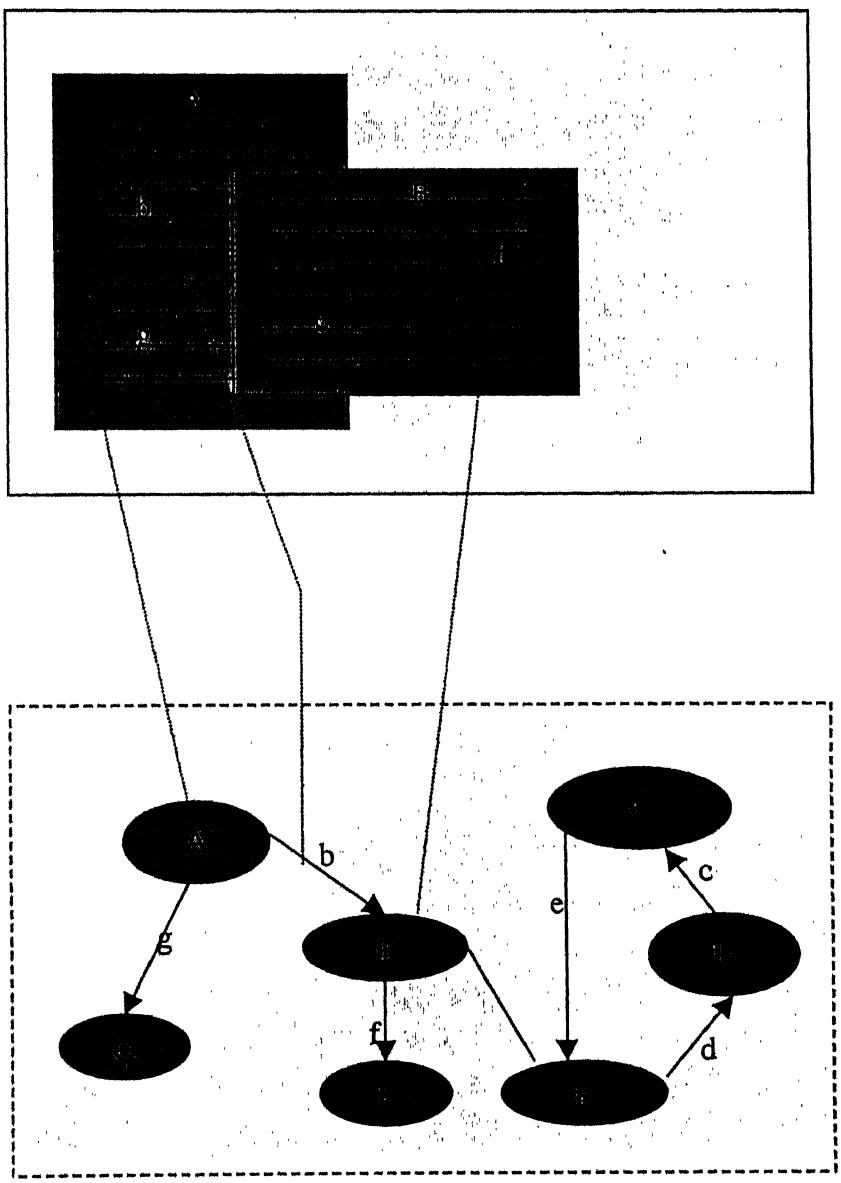
Some features of classroom instruction, which are important, are as follows:

i. A significant amount of material is involved in a course - typically a textbook worth of information.
ii. The material is taught over an extended length of time usually the order of several weeks. Though such material is often covered in much shorter time periods in "short term courses".
iii. The course usually has well defined objectives, one or more themes which are central and is carefully structured.
iv. A course is normally part of a curriculum. This implies that a course assumes certain pre-requisites or background. The course in turn is often a pre-requisite to one or more courses later in the curriculum.

In this thesis we discuss a conceptual framework for the first component in classroom teaching (i.e. lectures). Actually, the framework can be used to specify courses and even a complete curriculum. A complete case study for a course in programming through Java has been developed as an Example. A companion Mtech thesis has developed an authoring system, which can be used to author such courses. We refer to classrooms using such digital aids as digital classrooms.

HyperBoard Concept

Display Screen



HyperBoard Database

Figure 1

A Digital Classroom can be defined as a classroom where the Instructor takes the assistance of digital aids in imparting instruction. The Instructor develops the Information space and organizes it as per his line of thought. There are plug-ins attached that can be utilized by the Instructor with time to time as per need basis. E.g. While teaching Chemistry the Instructor may want to simulate the experiment by using tools that can do animations. A Mathematics teacher may want to demonstrate the functionality of the sine function by applying the same on some Data. The plug-ins can be MATHEMATICA or METLAB. A different Instructor can also utilize the original information space by carrying out additions and alterations to the Information space. In addition the Instructor can reorganize the Information space as per his line of thought.

It is expected that over a period of time the basic information space will be enriched and several courses corresponding to different instructors will be embedded in the same information space.

The authored material can be presented using a standard HTML browser so that the material also becomes available to students outside the class.

1.2 Hyperboard:

Hyperboard is the conceptual unit for creating and organizing the Information space. The word is derived from HYPERtext and blackBOARD. The Hyperboard carries information with it as a unit and has links to other Hyperboards. The information is written in a cross-referenced manner allowing the user to jump from one topic to another. A set of Hyperboards results in a lecture and a set of lecture result in the formation of a course. The Hyperboard concept is elaborated in Fig 1. The figure illustrates the way information nodes or Hyperboards are connected together. On the display the anchor nodes are visible that are connected together in the database. The links help easier and faster way retrieve the information.

The Hyperboards carrying information can have links, which can have different functionality than that of audio, video, image, and link to another Hyperboard. We can have typology of links, where each link can be defined to have a different functionality attached to them. E.g. in the thesis while discussing Java we can have keywords like Object, Class etc having links carrying functionality to have following features:

- Definition
- Figure
- Code
- Applet

1.3 Organization of the Report:

The rest of the thesis is organized as follows:

Chapter 2 discusses the digital aids for classroom instruction. It discusses the hypothesis that a key determinant of good instruction is proper organization of the material.

Chapter 3 discusses the Hyperboard concept in detail and the Hypertext paradigm, which forms the basis for it.

Chapter 4 discusses the Case Study, Java. The Java course material has been developed utilizing the authoring tool created.

Chapter 5 is the concluding chapter and discusses improvements in the tools and other aids for the digital classroom.

Chapter 2

Digital Aids to Classroom Teaching

The framework and the corresponding tools of this thesis are based on the following two hypotheses:

- Given the same basic information space the quality of instruction depends heavily on the way the material is structured and organized.
- The availability of context sensitive and subject specific tools which can work on the raw material of the lecture in different ways will significantly determine how well a student will understand the material. Such tools can include (drawing tools, graphing tools, symbolic evaluation tools, simulation tools etc).

2.1 Effective Teaching:

It is an observation that not all Instructors are alike. Some instructors are better than others are. Some possible differences (relevant to our hypotheses):

- The Instructor may not be able to organize his lecture in an effective manner. E.g. The Instructor while teaching any subject has to ensure that the lecture proceeds in a manner that it takes the reader slowly into the depths of the Information space. For Example while teaching Object Oriented Programming there are two ways the Information space can be organized:

    - The information space may be structured such that initially the various concepts are explained and then the same are elaborated with the help of examples. The information space may then explain the building blocks of OOP with reference to a language. This is the way it has been done in our case study.
    - The second way is to start with the building blocks of Object Oriented Programming language and whenever necessary link the HyperBoard to the concept which is relevant at that point.

  The first is a concept centric organization of the material with the programming language as a vehicle for representing or realizing these concepts in a concrete way. The second organization is language centric and the concepts are presented as abstractions of certain language features.

  The two approaches are quite different and have differing goals and it is not possible to substitute one for the other. However, the information space of both is largely the same, only the organization is very different.

- The lack of flexibility in organizing the lecture material. The instructor may want to refer to other course material to elaborate on a particular aspect may lack assistance. E.g. in Fig 2 observe that there are two courses which are interlinked. When multiple Inheritance is referred to in C++ , then to explain this the link goes to course material of another course. This kind of flexibility in linking material from two or more is useful.

- An Instructor may lack tools that operate on the data to assist him in the teaching process. E.g. an instructor teaching Mathematics while explaining the sine function will find a function graphing tool very useful to give examples and explain what is meant by a period etc. Similarly animation and simulation tools can be very important in instruction.

Cross-Referenced course material

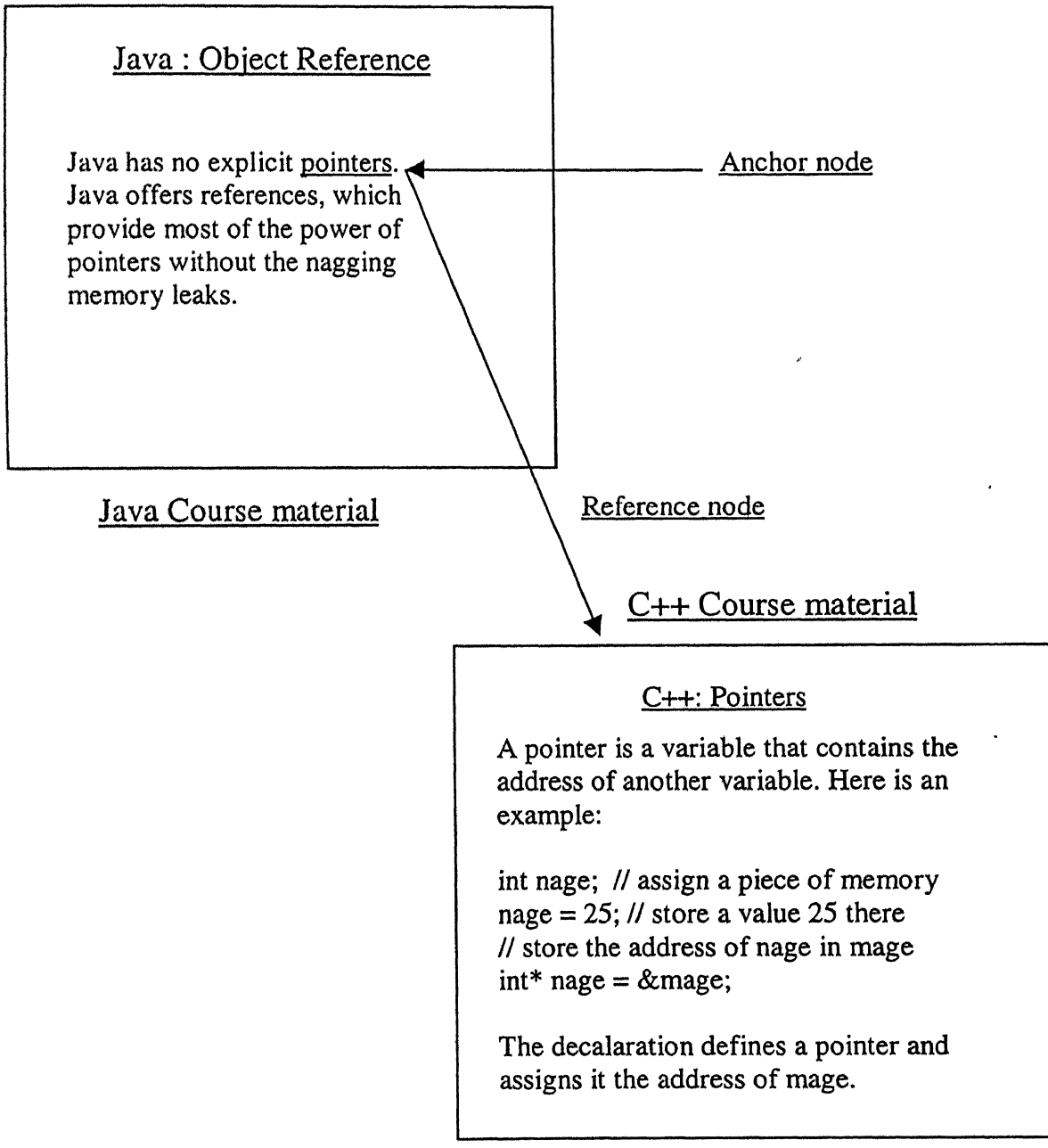


Figure 2

- The Instructor may lack material to elaborate on a particular topic. E.g. while teaching the interactive behaviour of an Applet the Instructor may want to demonstrate the same with an Applet itself. Similarly, programming concepts can be explained with the help of executable code, which may help in better understanding.

Digital aids in classroom instruction should overcome some of the above problems notably creating and organizing the information space and making available useful tools for better elaboration.

2.2 Digital Classroom and Hyperboard:

The Information space is, loosely speaking, the set or collection of all information on a particular topic or subject. The Information space carries all the relevant information, which is to be presented. A HyperBoard is the basic unit of information space. The set of all HyperBoards constitutes the information space.

The Organization and structuring of this Information space forms an organizational layer on top of this material. For Example in this thesis Information units or Hyperboards have been created for a course in programming in Java. The content of a lecture is a set of Hyperboards. The content of a course is a set of lectures. A lecture, the form in which it is to be delivered, is elaborated by building in suitable links which may also trigger the creation of more HyperBoards. The lecture proceeds by taking a path through the Information space, where the Hyperboards are visited as per the organization imposed by the Instructor. Similarly, a course (or a curriculum) can be visualised as a path through the Information space where now each lecture (or course) is encapsulated as a HyperBoard.

The Information space can have multiple courses (i.e. organizations/structuring) embedded into it. The Information space from other places / topics can be merged/accessed. This can be demonstrated in the Fig 2.

Here in the information space two courses are embedded, Java and C++. The organization of the Java lecture on Object reference links the Hyperboard to the C++ course material.

The Information space can grow incrementally. Different authors having different perceptions can add to the information space and also create a fresh organizational structure or modify an existing one.

Since multiple organizational structures can be embedded in the same information space it is possible to have different courses (e.g. elementary, intermediate, advanced) within the same information space. For example, in the present case the Java course material has 11 chapters in it. The course covers all classes of Java presently existing. The course can be structured to contain chapters 1,2,3,4,5,6,10 and section 1 of chapters 7,8 and 9 to draw up an elementary course on Java. The remaining information can be left out as it is meant for an advanced course. This way of structuring the Information will help an author to tailor the course to the requirements.

The HyperBoards in the Information space can have multiple links associated with the same element. Each type of link elaborates a different aspect of the link element. For example in the thesis, while discussing Java, we can have keywords like *Object*, *Class* etc having multiple links with the following types:

- Definition
- Figure
- Code
- Applet

Similarly in a Mathematics course we can have link types like:

- Definition
- Theorem
- Example

To summarize the HyperBoard framework provides the following:

i. Organization of the Information space in terms of chunks called HyperBoards.

ii. The same uniform framework from an individual lecture to a complete curriculum.

iii. The ability to embed multiple organizational structures on the same material to capture multiple authors or multiple levels of courses etc.

iv. Incremental changes.

v. The ability to linkup seamlessly with material from other courses.

vi. The ability to distribute over geographic locations.

Chapter 3

HyperBoard an Extension Of Hypertext

3.1 Introduction:

Traditional organization of textual information is Linear. The new extension of traditional text falls under the category of hypertext or nonlinear text. Hypertext, as defined by Ted Nelson is " a combination of natural language text with the computer capacity for interactive branching or dynamic display ... of a nonlinear text... , which cannot be printed conveniently on a commercial page.

With the advent of CD-ROM and videodisc technology hypertext is getting extended to the more general concept of hypermedia, in which the elements networked together can be text, graphics, speech, pictures, animation, film clips etc.

In this thesis we have introduced the concept of the "HyperBoard" which is a hypertext variant of Blackboard. First we briefly review Hypertext:

3.2 Hypertext : A Short Review

A Hyperdocument is a network of nodes and links. Each node is a chunk of information in one or more modalities (i.e. text, graphics, audio, etc). A node is similar to a page in a traditional book. A node can have one or more links to other nodes. The links are usually associated with some elements of the material, which make up the node. For Example in a hypertext document links could be associated with words or phrases.

The essence of hypertext is its ability to perform high-speed branching transactions on textual chunks. It can also be described as a computer-based medium for thinking and communication.

The traditional flat text binds us to writing and reading paragraphs in a linear succession. There are ways for signaling branching in the flow of thought when necessary: Parenthetical comments, foot notes, intersectional references, bibliographic references and sidebars all allow the author to give a related reference.

Hypertext allows the writer to make such references and allows making their own decisions about which links to follow and in what order. It does not force a strict decision about whether any given idea is either within the flow of a paper stream of thought or outside it. Hypertext allows annotations on a text to be saved separately from the reference document. The " linked-ness " of hypertext provides much of its power.

The most distinguishing characteristic of hypertext is its machine support for tracing of references. To qualify as hypertext, a system should require a couple of keystrokes or mouse movement to follow a link. The

link acts like " **magic buttons** " to transport the user quickly and easily to a new place in the hyperdocument. Properties of links are:

- They can connect a document reference to the document itself.
- They can connect a comment or annotation to the text about which it is written.
- They can provide organizational information.
- They can connect two successive pieces of text or a piece of text and all its immediate successors.
- They can connect entries in a table or figure to longer description or to other tables or figures.

The nodes contribute towards defining the operations that a hypertext system can perform. Nodes express a single concept or idea and are much smaller than traditional files. Hypertext invites the writer to modularize ideas into units in a way that allows:

- An individual idea to be referenced elsewhere and
- Alternate successors of a unit to be offered to the reader

The fact is that a hypertext node, unlike a textual paragraph, tends to be a strict unit, which does not blend seamlessly with its neighbours.

The Hypertext structure has been in use earlier:

- **The indexcard kind of manual hypertext.** This kind of manual hypertext uses 3 x 5 index cards for note taking. Note that the cards are often referenced to each other, as well as arranged hierarchically. The note cards being of smaller size modularize the notes into small chunks giving them an advantage. The user can easily reorganize a set of cards when new information requires

restructuring of nodes. A difficulty with this setup is finding a specific card if one has many cards.

- **The reference book:** The reference book is another kind of manual hypertext e.g. dictionary and encyclopedia. Each of these can be viewed as a graph of textual nodes joined by referential links. As one reads an article or definition, explicit references to related items indicate where to get more information about those items.

Bush, Engelbart, Nelson first described hypertext and they had the vision of hypertext as a path for human-computer interaction. There are four broad application areas for which hypertext systems have been developed.

- **Macro literary Systems:** The study of technologies to support large on-line libraries in which interdocument links are machine supported (E.g. Publishing, reading, collaboration etc).

- **Problem exploration tools:** Tools to support early-unstructured thinking of a problem when many disconnected ideas come to mind (E.g. authoring, problem solving, programming & design).

- **Browsing Systems:** Systems similar to macro literary systems, but smaller in scale (for teaching, reference etc).

- **General hypertext technology:** general-purpose systems designed to allow experimentation with a range of hypertext applications (for reading, writing, collaboration etc).

The importance of hypertext is that references are machine supported. Although traditional literature, like hypertext is richly interlinked and hierarchically organized its presentation is linear. There are problems with this kind of Organization:

- Most references cannot be traced backwards. A reader cannot easily find where a specific book or article is referenced in a document, nor can the author of a paper find out who has referenced the paper.
- As the reader winds his way down various reference trails he must keep trace of which documents he has visited and which he is done with.
- The reader must squeeze annotations into the margin s or place them in a separate document
- Finally, following a referential trail among paper documents require substantial physical effort and delays.

In contrast the problems with hypertext are:

- **Disorientation:** The tendency to loose one's sense of location and direction in a non-linear document.

- **Cognitive Overload:** The additional effort and concentration necessary to maintain several tasks or trails at one time.

A typical hypertext system has the following features:

1. The database is a network of textual and graphical nodes, which can be thought of as a kind of hyperdocument.
2. Windows on the screen correspond to the nodes in the database on a one-to-one basis and each has a name or title, which is always displayed, in the window.
3. Standard window system operations are supported: Windows can be repositioned, resized, closed and put aside icons. The position and size of a window or icon are cues to remembering the contents of the window. Closing a window causes the window to disappear after any changes that have been made are saved to the database node.
4. Windows can contain any number of link icons, which represent pointers to other nodes in the database. The link icon contains a short textual field, which suggests the contents of the node it points to.
5. The user can easily create new nodes and links to new nodes (for annotation, comment, elaboration etc) or to existing nodes (for establishing new connections).

6. The database can be browsed in three ways :

   - By following links and opening windows successively to examine their contents.
   - By searching the whole network (or part of it) for some text.
   - By navigating around the hyperdocument using a browser that displays the network graphically.

3.3 Summary of Hypertext:

- **Tracing References:** Machine support for link tracing means that all references are equally easy to follow forward to their referent or backward to their reference.

- **Creating New references:** Users can grow their own networks or simply annotate someone elses document with a comment.

- **Information Structuring:** Both hierarchical and non-hierarchical organizations can be imposed on unstructured information; even multiple hierarchies can organize the same material.

- **Global Views:** Browsers provide table-of-content style views, supporting easier restructuring of large or complex documents; global or local (node or page) views can be mixed effectively.

- **Customized Documents:** Text segments can be threaded together in many ways, allowing the same document to serve multiple functions.

- **Modularity of Information:** As the same text segment can be referenced from several places, ideas can be expressed with less overlap and duplication.

- **Consistency of Information:** References are embedded in their text and if the text is moved, even to another document, the link information still provides direct access to the reference.

- **Task Stacking:** The user is supported in having several paths of inquiry active and displayed on the screen at the same time, such that any given path can be unwound to the original task.

- **Collaboration:** Several authors can collaborate with the document and comment about the document being tightly interwoven.

3.4 New Possibilities for Authoring, Design, Reading and Retrieval:

Hypertext offers new ways for authors and designers to work. Authoring is a word and sentence level activity. The word processor is a good tool for authoring at this level. Authoring has the requirement to structure ideas, order the presentation and conceptual exploration. Authoring is the design of a document. The unit at this level of authoring is the idea or concept and this is supported by hypertext as the idea is expressed in a node. As new ideas/concepts are added the author develops them as independent nodes and then links them to existing ideas where relevant. Elaboration, comments, examples etc can also be created as independent nodes and linked appropriately.

Hypertext offers new ways for accessing large or complex information sources. A linear (nonhypertext) document can easily be read in the order in which the text flows in the book. The advantage of nonlinear text is the ability to organize text in different ways depending on different viewpoints. Another advantage is that in a hypertext environment we can suspend reading temporarily along one line of investigation and look into some other detail, example or related topic.

3.5 HyperBoard

3.5.1 Introduction:

HyperBoard applications require two main steps:

- Authoring
- Browsing

In a similar manner to that of creating textbooks, the first step is creating or authoring the information base. The authoring process requires creating and storing the information nodes and creating the relevant links. As an authoring tool:

- The HyperBoard should help develop front end for applications.
- It should cut the development time in doing so.
- It should increase programmer efficiency due to ease of use.

The browsing mode is meant for the teacher/student. It involves presentation tools for the user to scan through the information and search/utilize it.

3.5.2 Hyperboard Design Issues:

The various Issues involved in the development are:

1. Components of Information structure:

We need to develop the information in such a way that it supports the knowledge representation we want to promote. We need to make subsets of the information to convey our ideas. E.g. Information related to a given concept will relate to other information differently depending upon overall context.

The information is broken into atomic blocks called HyperBoards. Each HyperBoard contains a unit of information which normally is not meant to be broken down or divided any further. The HyperBoard may contain text, images, applets etc. Audio and Video are also possible but this has not been implemented yet. The second step involves the need to structure these nodes. To structure the information, first the atomic information building blocks are grouped into a pattern e.g. to embed a HyperBoard into another HyperBoard. After that we try

to interlink the HyperBoards. The anchors (text which is a link to another HyperBoard) in the HyperBoards form the starting point for links to other nodes.

2. Types of Structures and Topologies:

The organizational structures created during authoring fall under three main groups:

- Linear
- Hierarchical
- Network or Graph

- **Linear Structures :**

They maintain the sequential structure of the document. The access to information is in an order binding the series of nodes in a sequential manner.

- **Hierarchical Structure:**

This method uses a table of content or index and selects a point within the information base to start reading. Here a unique path exists between any two nodes. To reach another node from the current node, back up repeatedly to the unique parent of the current node until one reaches a node that is ancestor of the target node then go forward along the links to the target node.

- **Network or Graph:**

Here the HyperBoard system has associative links. The links allow the user to access the same information from different contexts. The links try to bind common or related concepts together in the information space. Here the topology supports nonunique paths between two hyperboards.

3.5.3 Developing a HyperBoard System:

Developing a Hyperboard system involves a number of issues for the whole process. A correct methodology is needed to have a good quality product. The authoring process for developing the information space can be of three kinds:

- Programming language based approach.
- Screen based approach.
- Information centered approach.

- **Programming language based approach:**

Here the applications are coded using a scripting language or programming language.

- **Screen based approach:**

Here we have every screen handcrafted and manually linked to build a HyperBoard application. The screen has a single chunk of information. The screens are linked together based on the needed structure.

- **Information centered approach:**

Here the content is obtained from existing information. The information is structured and stored in the database. The structuring involves first division into nodes and second linking up the individual nodes. It is after this that the information is presented on the screen.

During the development of the HyperBoard information space we need to address the following issues:

- Structuring information can be very time consuming when the space is large.
- Linking related information is based on knowing the location of associated information. This is the most challenging aspect for the author. As the information space grows it becomes difficult for the author to remember information contained in all the nodes. As size increases the cognitive overload is heavy.
- We need to develop an efficient way to reuse and manage the information once we have created the information space.

3.5.4 Navigating the HyperBoard

Differences in the way documents are to be browsed is built in as specified by an author on a document by document basis. HyperBoard based material can be structural in two ways:

- Logically linked information structure.
- Experience of an author of how the reader would browse the document.

Writing HyperBoard can be viewed as a form of programming. Specifying the structure of the document and the manner in which it is browsed can be done in a methodical way. There are different aids which help in navigating the information base:

- **BeatenBack mechanism :**

This navigational aid enables the user to return easily to places already visited, and include backup stack. The backup stack keeps track of the

node through which a user has moved, so that the steps can be retraced, bringing one back to the starting point.

- **Typed links :**

If links are classified by different types, the topology induced by links of any one type may be much simpler than the overall topology of the entire system. This will help the author to link up different Hyperboards and draw relevant information to elaborate. E.g. in the thesis while discussing Java we can have keywords like Object, Class etc having links carrying functionality to have following features:

- Definition
- Figure
- Code
- Applet

This way the confusion of criss-crossing through the network and getting lost is reduced.

- **Maps :**

A common feature of many hypertext implementations is a use of a map or geographical displays of nodes (often showing the titles, but with their contents hidden) and the links among them. Maps can be defined for large systems where the topology is constrained.

- **Navigational method utilized by us:**

Since HyperBoard texts are browsed using standard web browsers the browsing capability is restricted by what is available in them. It uses the beatenback mechanism. Here the browser keeps stacking up the nodes already visited. Through the usage of 'BACK' and 'FORWARD' buttons we can navigate the information space.

Case Study: Java

Chapter 4

4.1 Style Sheets:

This chapter discusses a case study in which a complete course in Programming using Java has been built using the authoring system built-in a companion thesis. The HyperBoard styles are created as required for presenting the information. The various types of HyperBoards depend on a Style Sheet. Styles are sets of properties or characteristics that govern the appearance of data within a document and modify them by editing a form called a style sheet. The style sheet can specify different text formats such as paragraphs, titles, font size, colour, headings links etc. The various types of style sheets created and used in this case are as follows:

- General Information Style Sheet
- Static code Style Sheet
- Index Style Sheet
- Question Style Sheet
- Answer Style Sheet

4.1.1 General Information Style Sheet:

This style sheet is meant to provide a style for each node to display the information. All Hyperboards which display information and cover textual matter, are governed by this Style Sheet. All topics, subtopics and other miscellaneous information, which impart some knowledge, are part of this style sheet.

4.1.2 Static Code Style Sheet:

This style sheet provides a template for the code of any program to be written. An Example, which is required to illustrate a topic, requires a demonstration in the form of some code. This style sheet helps create coded html documents so that the author only feeds in the program and the style is enforced.

4.1.3 Index Style Sheet:

This style sheets main job is to have a template for the index which leads to the index html document. In case a topic has more than one reference, the references are then indicated in the form of serial numbers as 1, 2 etc which are links to the relevant reference.

4.1.4 Question Style Sheet:

This style sheet provides a template in which the author can key in the questions, which form part of the questionnaire. This style sheets allows the author to key in the questions for various sections/chapters. A questionnaire is needed to test the comprehension of the subject by the author from the student.

4.1.5 Answer Style Sheet:

This Style Sheets main job is to help the author key in the answers of the questions, which were created, with the help of the Question Style Sheet.

4.2 Creating the Information Space:

The information base has been developed with following in mind:

- There should be clarity.
- There should be brevity.
- There should be modularity.
- There should be association in the topics clubbed together.
- There should be Examples both static and Dynamic to elaborate the topic.
- There should be Diagrams to help in understanding.
- Many Hyperboards are conceptual in nature and can be used even when the language is a different one (For Example C++).
- Java HyperBoards have been created separately.

4.3 The Information Space Organization

4.3.1 Design/Creation of Hyperboards:

The structure imposed on the information space is nonlinear. The various topics in the Java language have been created as Hyperboards. Each Hyperboard has been structured to contain the details pertaining to the relevant aspect of Java. There are links to elaborate on that particular topic. The design of the Hyperboard has been done such that different courses can utilize the same information space. To elaborate, the Hyperboards on *Object* and *Class* have been designed such that a person not interested in Java but wishing to understand Object Oriented concepts can also utilize the material.

Generally, the Hyperboards have been designed to constitute the following:

- Definition
- Static Example -- Code in Java
- Dynamic Example -- An Applet
- Figure

4.3.2 Organizing the Information Space:

The Organization of the Information space is possible in many different ways. There exists flexibility in organizing the same. Figure 3 will demonstrate that in the case study the following topics have been clubbed together to form a chapter:

- Extending Classes
- Final Classes and Methods
- Abstract Classes and Methods
- Object Class
- Interface

These Hyperboards exist as a separate entity in the Information space. They have been linked together to form a chapter.

The various chapters are formed in a manner similar to that shown for the chapter above and in the process 11 chapters have been created. The various chapters are all linked together. This has been illustrated in Figure 4.

Organizing the course:

Here we have created an advanced course on Java by covering all the topics which exist today. Using the indices we can tailor the course to our requirement. The Indices will help club the relevant Hyperboards into the course/lecture.

A main index will cover all the chapters in the course. Each Chapter in turn will have an index which will refer to the contents the chapter contains.

The advanced course comprises on following aspects of Java (Figure 6):

- Introduction
- Building Blocks
- Operators, expressions and control flow
- Classes and Interfaces

- Strings
- Exceptions
- Threads
- Input/Output Stream
- Packages
- Applets
- Misc.

An elementary course consists of following topics (Figure 7):

- Introduction
- Building Blocks
- Operators, expressions and control flow
- Classes and Interfaces
- Strings
- Exceptions -- Only index and section1
- Input/Output Stream -- Index, section1 and section2
- Applets

So we observe in the above case that we have an elementary course and an advanced course embedded in the same Information space.

- **Organizing the lecture:**

Similarly, course material from one course can be used in another course -- see Figure 2. The course material can be adjusted by an Instructor to move a topic from one place to another. For Example, in the case study Chapter 4 has the following topics:

- Extending Classes
- Final Classes and Methods
- Abstract Classes and Methods
- Object Class
- Interface

Here the *Object Class* can be moved to Chapter 2 that discusses on Objects and Classes. This way the author can tailor the chapters as per his line of thought.

4.4 The Information Space: Java

The HyperBoards in the Information space are presented in Appendix A. It has been done in a hierarchical manner as chapters and sections. This is entirely due to the limitation of presenting the material on a non-linear medium like paper. A schematic diagram illustrating the non-linear roadmap is provided for reference in figure 4. Each chapter consists of Hyperboards as shown in Figure 3 and Figure 5.

Chapter 4 Constitutents

(From the Information space these Hyperboards have been extracted to form chapter 4)

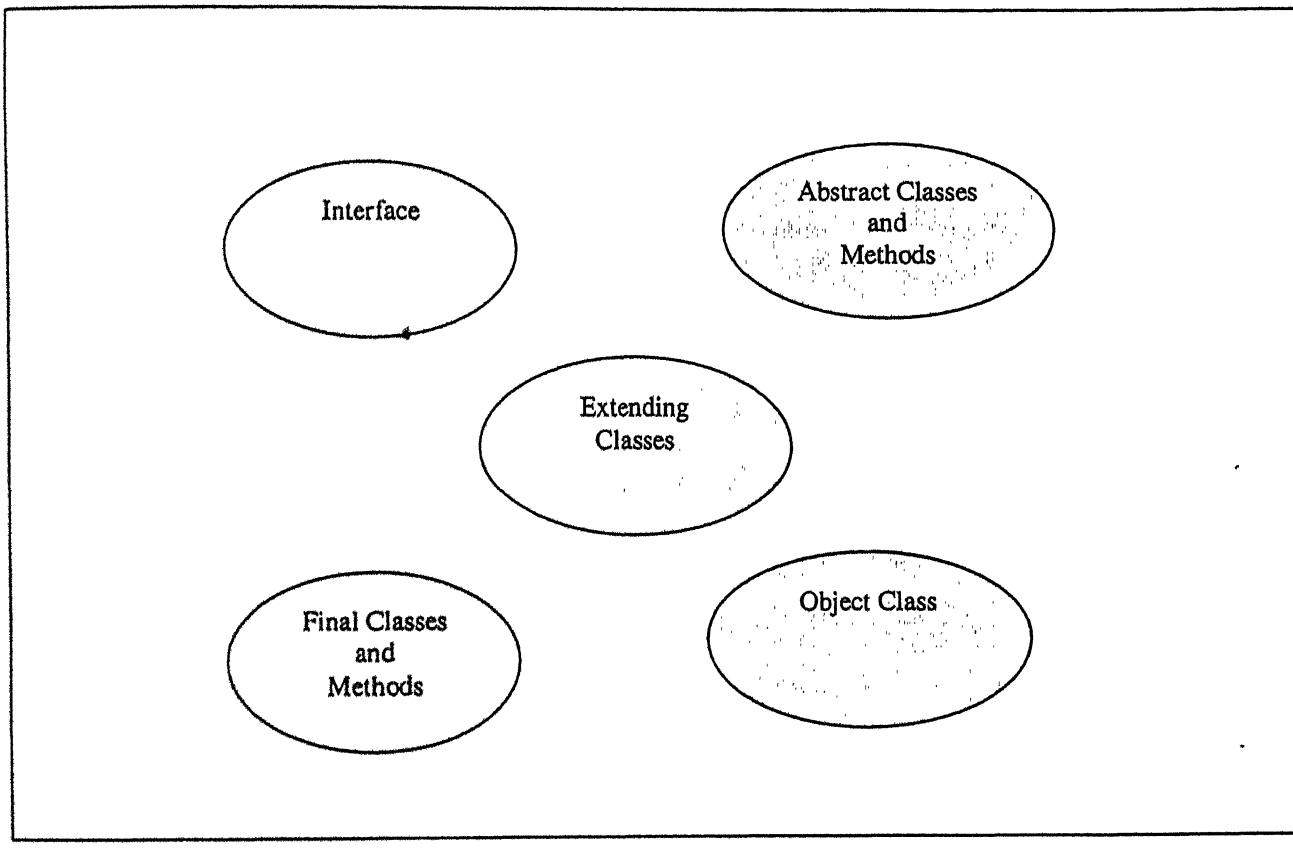


Figure 3

Information space Linkage

(A schematic diagram illustrating the chapters Linkage.)

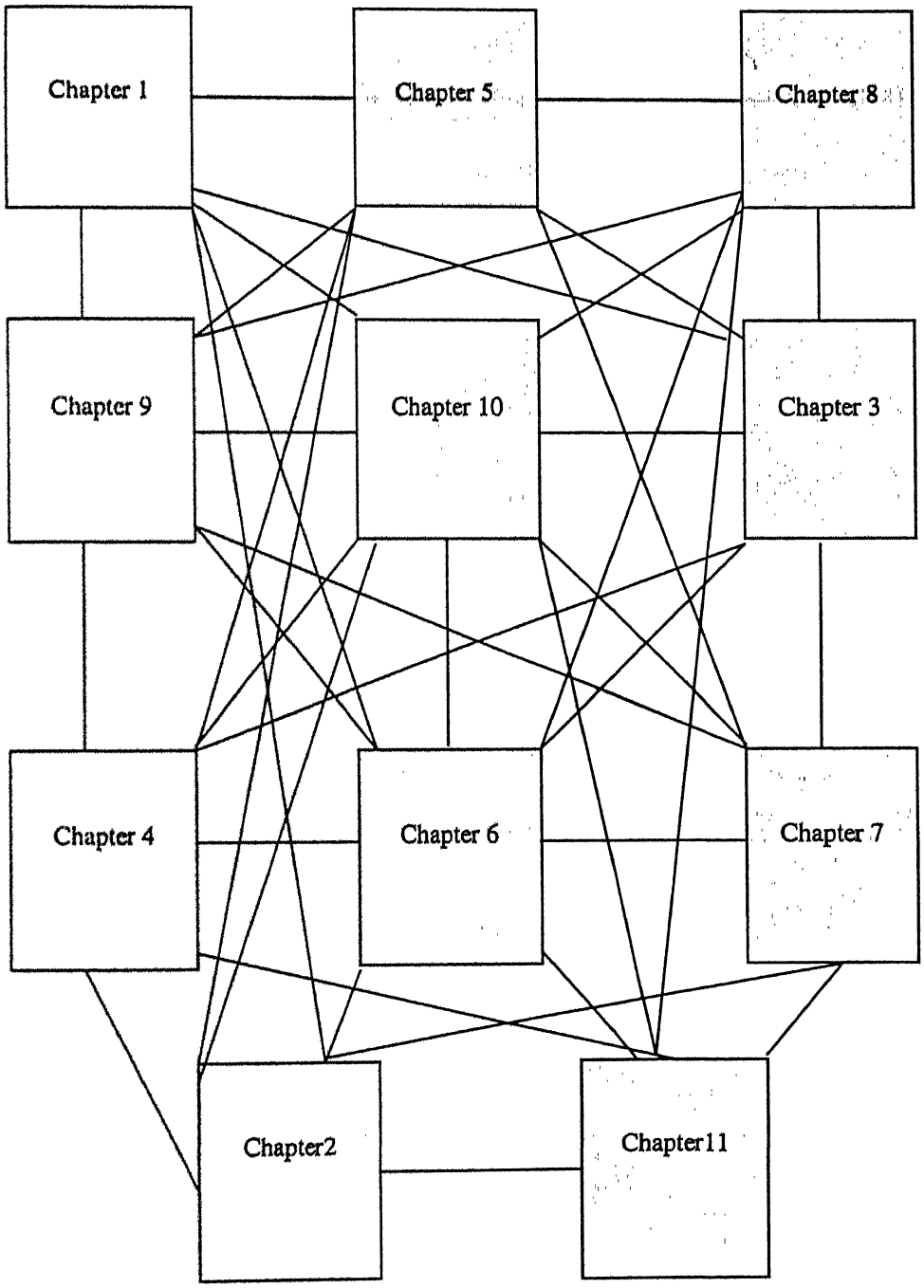


Figure 4

Information space Organization

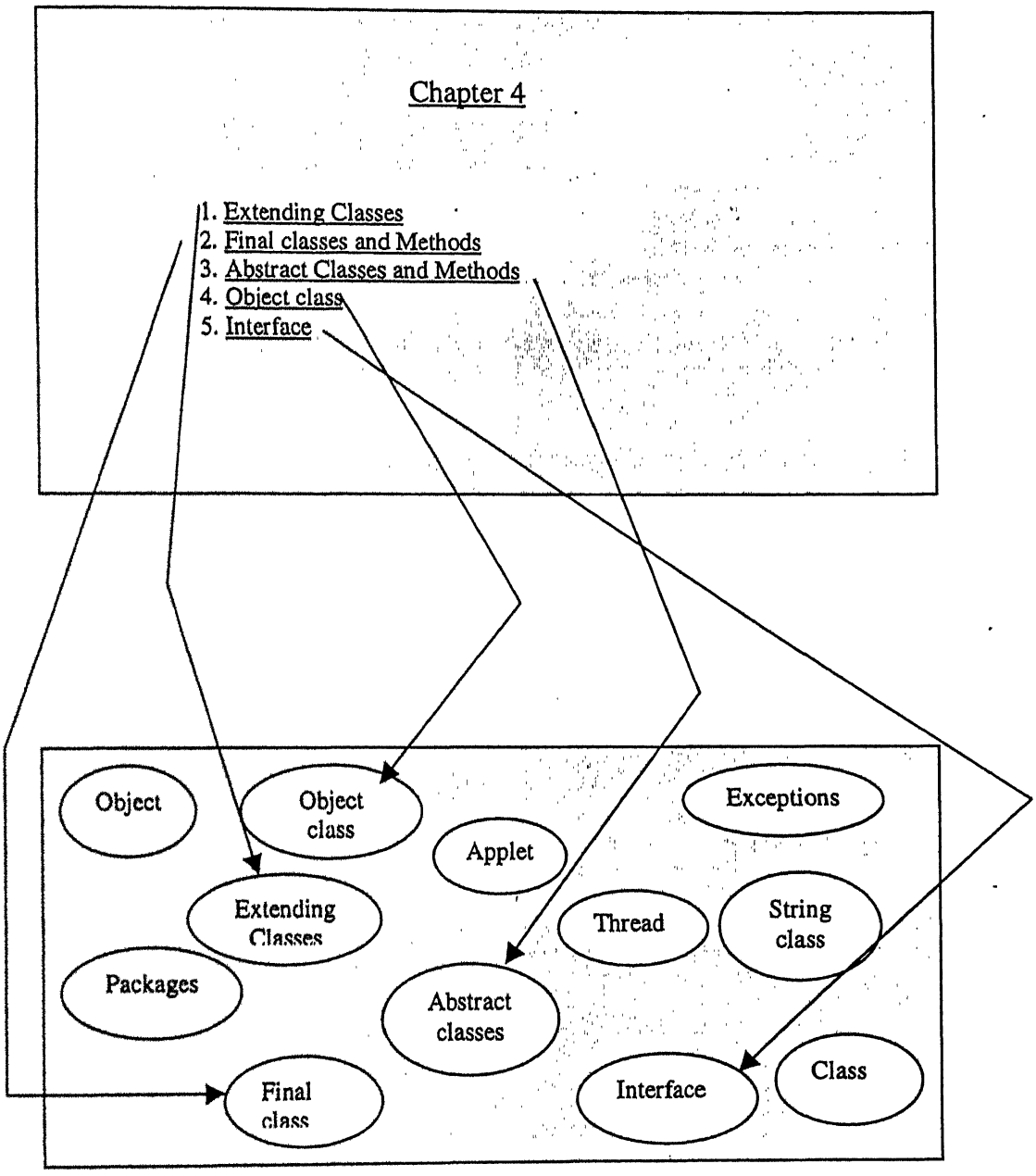
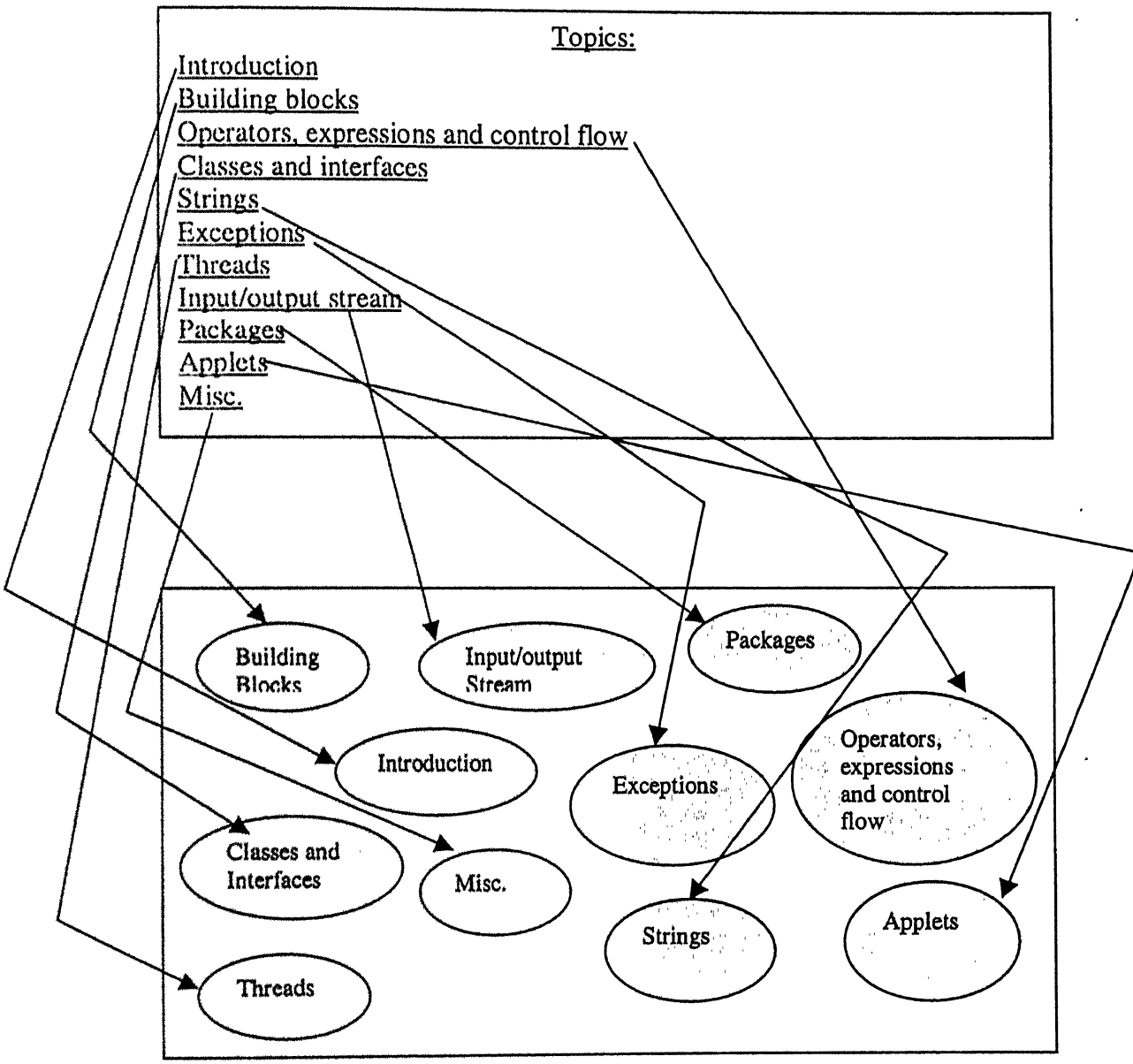


Figure 5

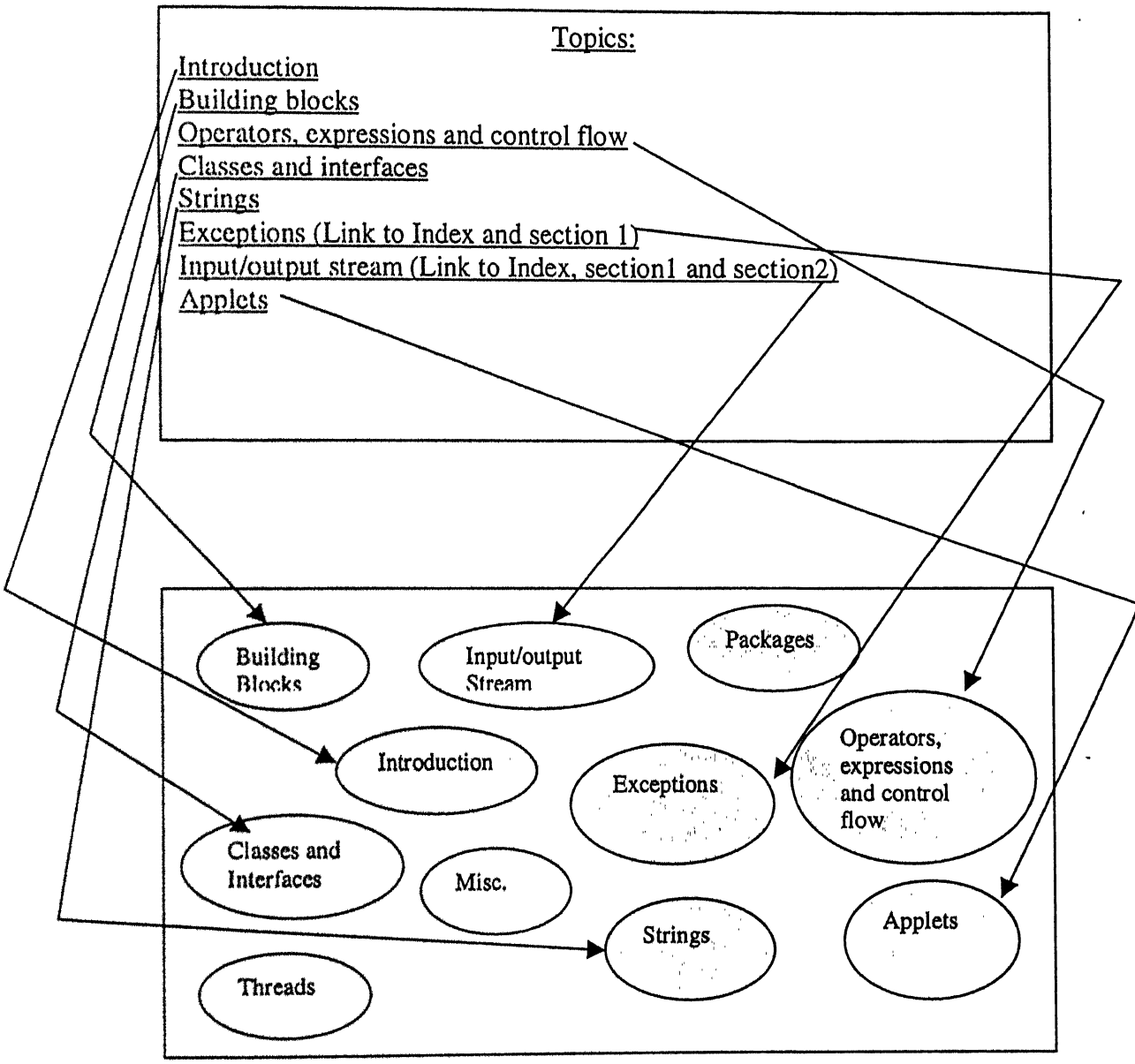
Advanced level course from the Information space



Information Space

Figure 6

Elementary level course from the Information space



Information Space

Figure 7

Conclusions and Future Work

Chapter 5

The work completed in this thesis has tried to explore how computers can help in classroom instructions. It has introduced the HyperBoard as a structuring unit and described. The HyperBoard brings the functionality of hypertext to a blackboard.

The HyperBoard framework allows an author to create information units as HyperBoard through user defined style sheets and then structure the information space through a user defined typology of links. We are conditioned to studying and teaching in a linear, hierarchical environment. So we lack instructional models and techniques which are intrinsically based on non-linear organization of information. The HyperBoard framework can be used to experiment with such models.

5.1 Future Work:

Areas that need more investigation are:

5.1.1 Authoring :

The current authoring tool that was developed in a companion thesis needs some improvements.

- **Typed Links:**

  There is need for supporting different types of links which are generic. For example an animation link, commentary link, example link etc.. Such link types should be sensitive to the type of the HyperBoard.

- **Better tools for data Organization:**

  Good organization of material often requires one to sometimes zoom in to get at more details while at other times one wishes to zoom out to get the birds eye view. So having various map like representations in which an author can zoom in or out would be useful.

- **Additional tools to be plugged:**

  Plugins are a popular way to add to the capability of browsers. Adding plugins specific to the subject will greatly enhance the explanatory tools available to a teacher. For example in mathematics it would help to have symbolic evaluations, graphing tools, animation tools etc.

- **Audio and Video Data:**

  Addition of audio and video will again give many options to a user to present material more effectively.

5.1.2 Browsing:

Currently, we are restricted by the functionalities of existing browsers as well as the current HTML standard which does not have good mathematics support. The present system of Navigation employed in our work is the Beaten-back path mechanism. Here we have used the Browser's capability in finding the path taken by the person browsing using the "Forward" and "Back" buttons. However, the browser needs to support other navigational devices which are more natural for HyperBoard.

5.2 Conclusion:

The thesis has made a beginning towards building a Digital Classroom through a case study of creating teaching material for Java

using the HyperBoard framework. The scope of work covered has been quite intensive. There are tools that have been created which would help the author in creating the database and the database created that is the Java Tutor is with the help of the tools created. The database has text material, figures, code in Java for Examples, Applets that provide interactive component to the material. Lot of sites were visited which had suitable material on Java and the Tutors so created were studied and different approaches towards the same aim were studied. Different authors had different lines of thought and some ideas have been borrowed and the shortcoming of their work was studied to find out a better solution towards building the database, which would be the one which would cater for everyone's requirement.

Bibliography

[1] Conklin, J. Hypertext and Hypermedia Hypertext: A survey and Introduction. IEEE Computers 20,9(1987), 17-41.

[2] Ginige Athula, Lowe David B. and Robertson John Hypermedia Authoring IEEE Multimedia Winter 1995.

[3] Yankelovich Nicole, Haan Bernard J. Meyrowitz Norman K. and Drucker Steven M. Intermedia: The concept and the Construction of a Seamless Information Environment IEEE Computers January 1988.

[4] Marshall A.D. and Hurley S. Interactive hypermedia courseware for the World Wide Web URL : http://www.cm.cf.ac.uk/Teaching/

[5]Dumont Raymond A. Teaching and Learning in Cyberspace IEEE Transactions on professional Communication Vol. 39 , No 4, December 1996

[6] Davis Stephen R. Learn Java Now from Microsoft

[7] Gosling James, Java Programming Language

[8] Hypertext'89 Proceedings Nov 5 - 8 Special Issue --SIGCHI Bulletin

[9] Web site of Sun Microsystems URL: http://java.sun.com

[10] Web site of GamesLan URL: http://www.gameslan.com

[11] Web site of Hobart and William Smith Colleges author David Eck URL: http://math.hws.edu/eck

[12] Java Tutorial author: Elliotte Rusty Harold URL: http://sunsite.unc.edu/javafaq/javatutorial.html

[13] Html Editor: writer Kris Nosack URL: http://lal.cs.byu.edu/people/nosack/

[14] Html Elements: Writer Michael J. Hannah URL: http://www.sandia.gov/

Appendix A

Java Course

The matter covered in various chapters is as follows:

A.1 Chapter 1

This chapter introduces the Object Oriented paradigm for constructing software. Java an Object Oriented Programming language is the language which will be studied as an example. This chapter covers the following:

- **Introduction to Object Oriented Programming:**

  This section defines the basic terminology used in Object Oriented Programming. This chapter covers the following:

  - Abstraction
  - Encapsulation
  - Modularity
  - Hierarchy
  - Typing
  - Concurrency
  - Persistence

- **Java Virtual Machine:**

  This section discusses how Java achieves platform independence through the use of a virtual machine. This chapter covers the role of a Compiler and an Interpreter. It goes on to show how Java uses both these features to produce programs which are platform independent with the help of Bytecode programs. The various transformations on the source program are described

here. This section also covers the features of the Java Byte code program and shows how Java is a language for network centric computing.

- **Java Vs C++:**

  This section covers Java and C++ along various dimensions like their Design Goal, Simplicity, Reliability and Robustness, Fast Translation and Efficient Object Code and Machine Independence.

- **Internet and World Wide Web:**

  This section covers the essentials of the Internet and the WorldWideWeb. These have been defined as they get mention in the text. This section covers the following applications of the Internet:

  - Telnet
  - Electronic Mail
  - File Transfer Protocol
  - Usenet News
  - Gopher
  - World Wide Web

A.2 Chapter 2

This chapter covers the basic building blocks of Java: Class and Object. This chapter covers the essential concepts on which the Java language is based. The various sections in this chapter are:

- Object
- Classes
- Access
- Memory Management

- **Object :**

  This section covers the definition of an Object. It talks about the relation of an Object with the class, creation of an Object, how an Object is lost, how an Object is accessed and the various methods and variables in an Object. There is also a static and a dynamic example (which is an Applet) to demonstrate the usage of the word Object.

- **Class :**

  This section begins by telling how class is the fundamental unit of Programming in Java. It gives the features of the class and its role, which include that class as an API, the relationship between two classes, how classes create new types and the role of classes in creating reusable code. There is also information on the properties of classes like naming classes, types of members and access to the elements of a class by other classes. There is a pictorial representation to elaborate on the relationship between Class and Object.

- **Access :**

  This section covers the various ways in which members of a class, sub class and other classes can be accessed. The different ways are as follows:

  - Public
  - Private
  - Protected

  Here we discuss members of different classes can access the members defined in some other class. By using the above access modifiers as prefixes to the name of the class we can allow or restrict access to members.

- **Memory Management:**

  Memory leaks are a serious problem in programming with language like C++. It has been a nightmare to programmers from the day they start. This chapter covers this most important aspect of how Java circumvents this problem. Here we do not face the problems of memory leaks or dangling references and overcome the problem of explicitly freeing the memory as in C++. There are ways in which Garbage collection is invoked which are:

  - Finalizing Object
  - Running the Garbage Collector

  Here we also discuss under what circumstances is the Garbage Collector is invoked. It can be explicitly invoked or start automatically. There is an Example to demonstrate it.

A.3 Chapter 3

This chapter includes the various operators, expressions and control flow elements used in writing programs. The Java program is written in Unicode a 16 bit character set. The various sections here are:

- **Comments:**

  Java has three types in which comments can be included in the source code. Their role is elaborated in this section.

- **Identifiers:**

  This section gives the constraints on identifiers.

- **Primitive Types:**

  This section covers the various primitive types, which exist, and the number of bits required for data elements of each type.

- **Literals:**

  This section tells about the various literals as each type in Java has a literal associated which can be an Object reference, Boolean, Integer, Floating-Point, Character or a String.

- **Declaring Variables:**

  This section constitutes the way in which we declare the variables in the program, which include access modifiers.

- **Array Variables:**

  This section describes the way in which the elements are declared in an Array are which are either primitive types or reference to other objects.

- **Operator Precedence and Associativity:**

  This section describes the way in which the various operators are given preference and how they are associated with each other.

- **Type Conversion:**

  Java is a strongly typed language. This section describes the way in which the various types are converted which can be explicit or implicit in nature.

- **Member Access:**

  This section describes the way in which we can access the members of some other class using the "." operator.

- **Arithmetic Operators:**

  This section describes the way in which we carry out arithmetic operations on Integers, Floating points and Strings.

- **Increment and Decrement Operators:**

  This section describes the way in which we carry out increment and decrement operations.

- **Relational Operators:**

  This section describes the way in which we use the equality and relational operators.

- **Bitwise Operators:**

  This section describes the way in which we use the bitwise operators like 'and', 'or' and 'xor'.

- **The Conditional Operator ? :**

  This section describes the way in which we use the conditional operator and its difference with the 'if' operator.

- **Assignment Operators:**

  This section describes the way we assign values to the variables using the "=" operator.

- **Statements and Blocks:**

  This section describes the types of statement in the language and how blocks are sets of statements.

- **if-else:**

  This section describes the most basic conditional that is "if - else" and how it can be used to check multiple conditions.

- **switch:**

  This section describes the way in which we carry out the check on multiple statements using "switch".

- **while and do-while:**

  This section discusses some basic looping constructs.

- **for:**

  This section describes how looping can be carried out inside the block till the Boolean expression is true. It can be used to create an infinite loop.

- **Label , break and continue:**

  Java does not support the go-to statement. This section describes the way in which we can use these constructs to transfer control.

- **return:**

  This section describes the way we use the return statement to come out of a method and how it is different from the way it is used in C.

- **go-to**

  This section describes the usage of the go-to command and where it is used and how such applications are handled in Java.

A.4 Chapter 4

This chapter describes two important constituents of Java, which are Classes and Interfaces. Classes form the fundamental unit of programming in Java and Interfaces play a role similar to classes but with some differences. The various sections in this chapter are as follows:

- Extending Classes
- Final Classes and Methods

- Abstract Classes and Methods
- Object Class
- Interface

- **Extending Classes:**

  Classes already declared can be extended to form subclasses. We extend classes to use the members of the classes already existing and add more functionality to the classes. The original class becomes the base class and the extended class becomes the subclass. This chapter elaborates the way in which we use constructors in the extended classes, the way we override the functions already declared in the base classes. There is also discussion on Polymorphism, inheritance and its realization in Java.

- **Final Classes and methods:**

  Java does not have a constant construct explicitly but the term "final" plays a role similar to that of *constant*. By declaring a method final means that no extended class can override the method to change its behavior. The method becomes the final version. The class so declared cannot be subclassed or extended. This affects the security of the class, it simplifies optimization and the type checking becomes faster.

- **Abstract Classes and Methods:**

  Abstract classes are classes that define part of the implementation. Here the declaration exists of the classes that are defined in other classes. Methods not defined explicitly are declared abstract and so are the classes having abstract methods. This section also gives the features of the abstract classes.

- **Object Class:**

  Object is the super class of all classes. All classes extend the object class directly or indirectly. All classes inherit the methods of the object class. The methods defined in the Object class are of general nature and so cater for all classes which are being defined.

- **Interface:**

  An Interface is a collection of abstract methods. Java does not support multiple Inheritance. An Interface adds multiple Inheritance to Java. An Interface is more a design issue and has nothing to do with implementation.

An Interface constitutes abstract methods, which are inherited by the class using the implement keyword. The following is covered in this section:

- Multiple Inheritance
- Extending / Implementing Interfaces
- Usage Of Interfaces

The second section describes the way Interfaces can be extended and also describes how Interfaces can extend two or more interfaces unlike in classes where only one class can be extended. The third section describes places where Interfaces can be used.

A.5 Chapter 5

This chapter describes the way Strings are handled in the Java language. Java strings are treated as objects. Java has two string classes namely:

- String
- StringBuffer

These two classes have different functionality. The String objects are read-only. They cannot be used to carry out operations on string in which the string changes. There is another class called StringBuffer. This class is used to provide facilities to mutate the strings. These two classes have different features and are given in detail under following sections:

- String Class
- StringBuffer Class
- String Conversions

- **String Class**

  The String class in Java is a read-only class. There can be no modifications done on objects of this class. The string represents a Unicode character string. This section describes the way in which the strings are created and the various methods that exist in this class. The various methods allow read-only operations on strings.

- **StringBuffer Class**

  To build and modify a String we need the StringBuffer Class. Please note that StringBuffer does not extend String Class or vice-versa. The

StringBuffer class helps carry out mutations on strings. Under this section following is covered:

- StringBuffer Class Constructors
- Modifying StringBuffer Objects
- Getting Data Out of the StringBuffer Object
- Capacity of StringBuffer
- Comparision of StringBuffer and String Object

- **String Conversions:**

  This section describes ways in which we can convert the various primitive types in Java. Java has a conversion convention such that the type being converted to has the method that does the conversion. There is a tabular representation of ways we can convert the different types to String and from String.

A.6 Chapter 6

This section describes the way in which errors are handled in Java. Java uses Exceptions to handle errors. There is a class called Exception that helps the programmer overcome the task of handling errors in programs in a clean manner. The term exception is shorthand for the phrase "exceptional event". An exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions. An Exception can find out the errors which can pertain to the internal state of an object, hardware crash, array out of bound etc. The various sections in this chapter are as follows:

- Creating Exception Type
- Throwing an Exception
- Catching an Exception
- finally block
- Usage of Exception

- **Creating Exception Types:**

  This section describes the way in which Exceptions are created. Exceptions in Java are Objects. Any class throwing an exception Object will have the class extend the word throwable in front of it. This section describes how the error check is carried out in two phases. In the first phase there is a compiler check. These kinds of errors are called Checked Exceptions. The second phase constitutes the run-time check. These are

the unchecked Exceptions and are also called as "Errors". The advantages of the Unchecked Exceptions or Errors are discussed in this section.

- **Throwing an Exception:**

  This section describes the way in which an Exception is thrown. Creating an exception object and handing it in the runtime system is called throwing an exception. In Java, an Exception occurs when the program executes the *throw* Keyword. We always throw an object of class Exception or some subclass of Exception. A method not having any exception means that no exception may be thrown. This section also describes about 'throws'. Java requires the declaration of checked exceptions that methods throw. The checked exceptions a method can throw are declared with a throw keyword.

- **Catching an Exception:**

  This section describes how the Exceptions are detected and are caught. The try block is the one where we carry out a check to see if any exception has occurred. Incase an error is caught then the same is thrown to the catch block where the error is handled. Here in Java we have a graceful way to overcome the sudden break of programs that take place due to errors. The catch block allows E.g. the resources of the system to be released before the program can stop.

- **finally block:**

  This section describes the role played by the *finally* block in the Exception mechanism. Finally block is part of the try block which is executed whether or not an exception is thrown. The finally block of a try statement is executed by the runtime system before control exits the method. The area, where *finally* plays a major role are elaborated in this section.

- **Usage Of Exception:**

  This section discusses the usage of Exceptions which range from reducing programming complexity, making readable code, not ignoring errors, provision of alternative error paths and storing more information about the error.

A.7 Chapter 7

This chapter describes one of the most important features of the Java programming language --Threads. A single sequential flow of control within a process is called a thread. In general programs execute one step at a time and are thus called single-threaded programs. There are cases when there is a requirement to execute multiple steps at a time. This is achieved by multithreading. Java supports multiple threads to execute simultaneously. The various sections described in this chapter are:

- Threads Constituents
- Synchronization
- Thread Scheduling
- Suspending & Terminating Threads

- **Thread Constituents:**

  This section describes the various constituents of a thread. Threads in Java are defined in a class. The thread class is system-independent. The various sections under this head are as follows:

  - Thread Body
  - Thread State
  - Thread Priority

  These sections are described in detail and there is a static example and a dynamic example to elaborate the topic.

- **Thread Synchronization:**

  This section describes the way multiple threads are synchronized to work in harmony. Synchronization forces execution of two threads to be mutually exclusive in time. This section describes the situation under race conditions. This section describes following:

  - Synchronized Methods
  - Synchronized Statement
  - Wait and Notify

- **Thread Scheduling:**

  This section describes the way multiple threads are scheduled. When there are multiple threads, then they are given priority and under circumstances a thread of higher priority is given preference or a lower priority thread.

- **Suspending and Terminating Thread:**

  This section is divided into two parts wherein the first part describes why and under what circumstances the threads are suspended. There is an Example to demonstrate the same. The second part describes how the threads can be terminated.

A.8 Chapter 8

This chapter describes the way input/output operations are handled. Java I/O is defined in terms of streams and a stream is a flowing sequence of characters. The input / output sources comprise of Files, Memory and String. All classes under this head are stored in Java.lang package. The various sections under this head are as follows:

- System class
- Input & Output Stream
- Types of Input & Output Stream
- Filter Stream

- **System Class:**

  This section describes the system class, which plays a major role for putting the data out and taking it in from various types of streams. All the input and output operations are handled by this class. This section is described in two heads:

  - System.out
  - System.in

- **Input and Output Stream:**

  This section describes how the input and output operations are carried out with the help of streams. The Java.io package has two classes the input

stream and the output stream. These two classes form the origin of other classes, which implement input and output operations.

- **Types of Input and Output Streams:**

  This section describes the various types of input and output streams, which exist in the main input and output stream classes. The various types of stream are divided under two main types:

  - Simple Input and Output Stream
  - Filtered Stream

  The various types of stream classes with different functionality are listed in these two sections.

- **Filter Stream:**

  This section describes the Filter Stream. The Filter streams are abstract classes representing byte streams with some filtering operation applied as bytes are read. There are two Filtered Stream classes namely the FilterInputStream and the FilterOutputStream.

A.9 Chapter 9

This chapter describes packages. A package is a loose affiliation of classes. Every class belongs to a package. Every class is added to the package when it is compiled. If the package does not exist, it is created when it is compiled. Java programs are organized as sets of packages. The members of Packages consist of all related classes, interfaces and subpackages. The various uses of package are listed in the beginning. This section is described under the following heads:

- Package Naming
- Package Access
- Package Contents
- Package and File System

- **Package Naming:**

  This section describes the way packages are named. Like classes and Interfaces the naming of packages should be unique. There is a typical

style by which the packages are named and is thus described in this section.

- **Package Access:**

  This section describes the way classes and Interfaces in packages can be accessed. There is a code of conduct where in the different members of classes depending on access rights can access the members of the classes in different packages.

- **Package Content:**

  This section describes which classes and Interfaces should be put in one package. A source file not provided with any package declaration goes to an "unnamed package". Generally all related classes and Interfaces go into one package.

- **Package and File System:**

  This section describes the relationship which the FileSystem and package share amongst themselves. There exists a direct mapping between the Package and the Operating System directories.

A.10 Chapter 10

This chapter describes the most important feature of Java for Internet programs, the applets. Java is fast becoming an integral part of the World Wide Web, the most interesting and fastest-growing Internet mechanism, in the form of "applets" that can be directly embedded into Web pages. The various topics covered under this chapter include the following:

- Applet and its Hierarchy Chart
- Phases of an Applet
- Various Kinds of Methods in an Applet
- Utilizing Applet's Capabilities
- Java Application Program

- **Applet and its Hierarchy Chart:**

  This section describes the hierarchy chart of the Applet and indicates its exact position. All Applets are part of the Java.applet.Applet package. This section is further divided under two heads:

  - **Hello World Example**

    - Create a Java Source File
    - Compile the Source File
    - Create an HTML File that Includes the Applet
    - Run the Applet

  - **Critical Analysis Of The Example**

    - Import Statement
    - Subclass formation
    - Implementing Methods
    - Running the Applet
    - No main method
    - Event Driven

- **Phases of an Applet:**

  This section elaborates the various phases, which an Applet undergoes. This section describes the following:

  - Loading the Applet
  - Leaving / Returning to the Applet Page
  - Reloading the Applet
  - Quitting the Browser

  What happens to the Applet under the above circumstances is elaborated in the sections listed above.

- **Various kinds of Methods in an Applet:**

  This section describes the various kinds of methods, which exist in the Applet. Each type of method has a different functionality associated with it and the same is elaborated in this section. The various kinds of methods listed here are as follows:

  - Basic Methods

- Methods for Components
- Methods for Drawing
- Methods for Event Handling

- **Applets Capabilities:**

  This section describes the various capabilities and restrictions which are applicable when we use an Applet.

- **Java Application Program:**

  This section describes Java Application programs. The main difference between the Java Application program and an Applet is that the Application program is a stand-alone program. To execute a Java Application program this section explains the following steps involved in the process:

  - Create a Java Source File
  - Compile the Java Source File
  - Run / Execute the File

  There is critical analysis done of a Java Application program which covers the following:

  - Defining a class
  - Main method
  - Comments
  - Using Classes and Objects

A.11 Chapter 11

This chapter compares Java with other languages with special emphasis on C++. The various sections covered under this topic are as follows:

- Templates
- Pointers
- Destructor

- **Templates:**

  Java does not have any template construct, which is there in the C++ language. Java obviates the need for templates. Here, in Java all classes are based on Object. There is an Example to demonstrate how templates can be implemented in the language.

- **Pointers:**

  In C++ another important feature is the pointer. This is not present in Java. There is lot of limitation while using the pointers and so Java has done away with it. In place of pointers Java uses object references and tries to implement the functionality provided by pointers.

- **Destructor:**

  This section describes about the role of destructor and how Java implements the same functionality without them. In C++ it was observed that the system resources are consumed through the constructor. When the resources are to be given back to the system then one uses the Destructor, which was used to release the resources. In Java garbage collection removes the need for destructors. Its primary role is to look for Objects who have no reference and to put them back in the heap.